```
 * Data Kult *            °²°²°²°²°²°²°²°²°       * Kryptic Night *
   Lord Logics            ²°   Raising   °²         Bounty Hunter
   Shadow Walker          °²    Hell     ²°         Nacht Habicht
    - S M C -             ²°  with Unix  °²          - S M C -
 Realm of Infinity        °²Kryptic Night²°        The Viking's Den
   (503)629-0814          ²°²°²°²°²°²°²°²°²         (408)867-1224
     SMC Home                 - S M C -              Western Dist.
                            Production # 3
 ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
```

I - Introduction

This file will describe several ways to cause mischief on a Unix system. Like the other SMC Productions, I will try to present the information at a beginners level. However, all levels of hackers should benefit in some way from the information contained within. And now... on with our show...

II - How To Fill a Hard Disk

There are several ways to cause havoc by filling up a systems hard disk. Filling up a hard disk will make it so that the system cannot create the temporary files vital to it's efficient use. It will also cause other problems, such as a person trying to save a 10 page financial report, and finding that there is no room for it. Also, if the HD is full, the system will not run properly. You will be bombarded by a continuous stream of 'write failed, file system is full'. Over all, this is a very good way to piss people off.

Step One

Create the following file with the 'ed [filename]' utility under the bourne shell, or the 'edit [filename]' under the C shell. The filename can be whatever you want, here I will call it 'hah1'. Only type in what is contained within '[]'s, the other text is what the system will send to you.

```
$[ed hah1]
0
*[a]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[echo Hahahahahahahahahahahahahahahahahahahahahah!!! >> -fucku!]
[nohup hah1 &]
[^C]
*[w]
754
*[q]
$[chmod +r+w+x hah1]
$[nohup hah1 &]
1234
$
```

This will create a file called '-fucku!'. Files beginning with a '-' are very difficult to delete, as when you try to do a 'rm -fucku!' <rm - remove file> It interprets the '-f' as an option, it tries then to force delete the file 'ucku!'. As you can imagine.... this wouldn't quite work. The text after the echo can be anything you wish, I just used a sample text that is quite pointless and takes up space. The numbers represent the file size, and process number, they will be different on your system.

The file will add the text from the echo statement to the file '-fucku!'

until it reaches the 'hah1 &' command, which will make it start over again. This is an endless loop. For as long as you are on-line, and their are processes left, the file will continue to add to the file. This is a very slow method, but it's easy if you are starting from scratch. If you get a message such as 'cannot fork hah1: process terminated' that means that the loop is taking up so much memory that the system can no longer continue with that job. Don't worry, it will settle back to normal after all the processes are eventually killed, if it does, continue to run the file in the background until you have a '-fucku!' file that is about 100-200k long, this will allow us to progress to our next step.

The command 'nohup hah1 &' tells unix to continue to run the 'hah1' in the background, even after you hangup. This means you can run the program, hang up, and call back. This function will only work under the bourne shell. If you have a prompt of '$', then you are using the bourne shell. This function will become exceedingly useful when we start with the next step.

The command 'chmod +r+w+x hah1' will make the file readable, writable, and executable by you. This string may or may not be necessary on the system you are using. If you get a message such as 'hah1: Permission Denied' than you will need to use the chmod command. And now onto the next step...

Step Two

We will now explore the ever powerful 'cat' command. The 'cat' command is the equivalent of the MS-DOS 'type' command. We will use a function of the unix system called redirection that will allow us to 'cat' files into each other. This will cause the source file to be copied to the end of the destination file, I'm sure you're beginning to see the mischief you can cause with this.

To begin with, create a file called '-fucku2' the same way you created the '-fucku!' file. Try to run the 'hah1' program until the new 'fucku2' file is around 100-200k also. This isn't absolutely necessary, but it's helpful and saves some time.

Next, create the following file with the editor <'ed' or 'edit'>. I will call it 'hah2', but you may call it whatever you wish.

```
$[ed hah2]
0
*[a]
[cat -fucku! >> -fucku2]
[cat -fucku2 >> -fucku!]
[no
hup hah2 &]
[^C]
*[w]
61
*[q]
$[chmod +r+w+x hah2]
$[nohup hah2 &]
7049
$
```

What we've just done is create a very short, and very nasty, program that can fill 20 megs in under 5 minutes. The file when run will add the contents of '-fucku!' to the end of '-fucku2', and do the reverse. This means that when you have two files of 100k to begin with, you will get the following results after every completed loop...

```
-fucku!  ..  -fucku2  ..  -fucku!  ..  -fucku2
 100k    >>   200k    >>    300k   >>    500k
 700k    >>   1200k   >>    1900k  >>    3100k
```

As you can see, the file grows VERY quickly. Set it up in the morning before school, come back and the HD should be completely full. You may wish to also run multiple write processes, just to confuse the system. If you do, rename the files to something appropriate, but maintain the base content. If you do it in several directories, the sysop will have to do some serious cleaning to get rid of it.

Step Three

Sit back and laugh. If you wait awhile, in approximately 30 minutes, the average 40 meg hard drive will be full. I've tested this method on several systems, even an ancient VAX, and the results were more or less the same. The sysop, or any other user, will be able to write anything onto the system until this problem is resolved. Many programs need to create temporary files to even operate. These programs are now completely unusable, except for the few that save to memory. To delete the files, the sysop will have to do one of several things, all of which are very unpleasant. And now for the next lesson...

III - Mischief

This section will describe a couple of ways of perpetrating mischief on a unix system. These ideas are for the most part harmless, but can definitely piss people off. The idea of a continuous subdir was molded from one presented by Shooting Shark.

Idea #1

This method will create an endless amount of directories under a the current directory. Create multiple files with different name and directories to really annoy the 'sop. Type the following to accomplish this.

```
$[ed sub1]
0
*[a]
[mkdir -FuCkU!1]
[chdir -FuCkU!1]
[/xxx/xxx/sub1 &]
[^C]
*[w]
69
*[q]
$[chmod +r+w+x sub1]
$[nohup sub1 &]
7099
$
```

This program will create a directory called '-FuCkU!1', change to that directory, then create another one under the first one, and so forth. It is an endless loop, and will continue virtually forever. The third line of the program contains a string '/xxx/xxx/sub1 &'. You will need to fill in the x's with your current directory. To find out your current directory type 'pwd' this will print a string telling which directory you are in. Fill in the x's with this data. The rest of the program you should be able to figure out by now. Try it, you'll like it.

Idea #2

So, you've seen someone on the system that you really don't like? Or do you just want to piss someone off? This methods for you. This method will describe a way to send out data to another user, or terminal. Here is what you will want to type to create a file to anger the other user.

```
$[ed beep]
0
*[a]
[echo ^G ^G ^G ^G Wheee!!! ^G ^G ^G >> /dev/xxxx]
[nohup beep &]
[^C]
*[w]
25
*[q]
$[chmod +r+w+x beep]
$[nohup beep &]
8002
$
```

Fill in the '/dev/xxxx' with the terminal you want to annoy. To find out the terminal of the person you want to fuck over, type 'who' it will print out something like this....

```
$[who]
guest    ttyd0   Nov 30 19:06
root     console Nov 30 19:20
Bendover ttyd5   Nov 30 18:45

$
```

The first column is the name of the user, the second column tells us what terminal they are logged on as, and the third states at what time they logged on. The second column is what we need right now. Fill in the x's with the terminal that you wish. If you wanted to bother the root, you would type '/dev/console', to bother guest type '/dev/ttyd0'. To bother more than one terminal, just add another line after the first 'echo' statement with a different terminal identifier. With the 'nohup' command, the computer will send a continuous outpouring of beeps until he logs off or reboots the system. Try it on the terminal you are logged on under to see exactly what it does.

ÄÄ

IV - Conclusion

These projects should be enough to get you started on your road to Unix Hell. With a little experience you will be able to think of new ideas that will alloy you access to the systems hidden features and assets. I will release other files on Unix in the near future, possibly one on basic Unix hacking, FTP, UUCP netting, or any number of other Unix related concepts. If you are interested in learning more on Unix, you can contact me on the systems at the top of the file. Thus concludes one dark Kryptic Night...

V - Bibliography and Suggested Reading

Unix Use and Security From the Ground Up: by the Prophet in 1986
This is probably the BEST file I've ever seen on the subject of Unix. It is written for the beginner, and contains valuable information for the advanced user. The Prophet became a member of Lod/H and is currently serving a sentence of 20 months in relation to the big Lod/H bust of '90.

Articles on unix trojans and mischief: by Shooting Shark
Shooting Shark presents some interesting information on various ways to commit havoc on Unix systems. You can find most of his essays in both Phrack and Lod magazines.

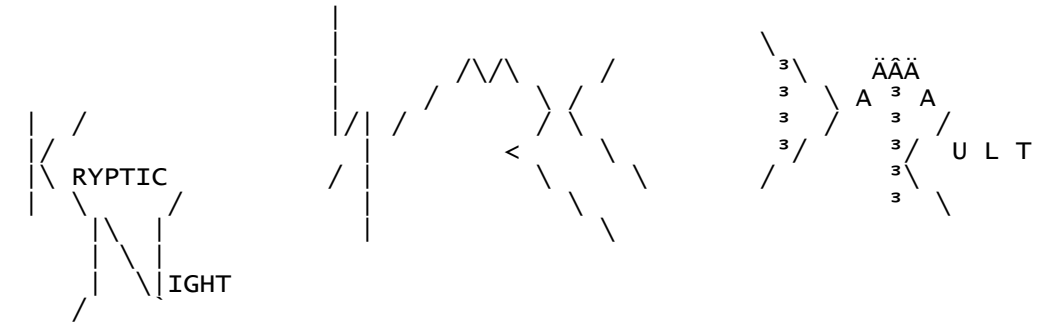
Lod/H Tech Journals
The Legion of Doom/Hackers are perhaps the most skilled and knowledgable hackers in the underground society. Their 'Tech Journals' describe almost anything you'd ever want to know about illegal activities.

Phrack Magazines
Phrack is also one of the best sources for information on a multitude of subjects, ranging from social engineering, to carding, to making explosives. For those with free time, download all of the 32 articles released to date.

Creating Users on Unix
This was my second text file release. It tells how to create new users on a Unix system using the root account. It is told for beginner and advanced hacker alike.

VI - Greets

Heh, Data Kult, when you gettin' Kelsea's phone number?
Bounty Hunter, cool new software, hope you can work out the bugs.
Lord Logics, ega STILL? Come on! Get with it!
Scooter, chill with the 800's
Oolon, get Entropy back up!
Digital Derelict, Jerusalem is nothing.... you're going down... soon



- Kryptic Night, Data Kult, Lord Logics, Shadow Walker, Bounty Hunter -
Nacht Habicht